

MEMORY INTERFACE AND METHOD OF INTERFACING BETWEEN FUNCTIONAL ENTITIES

Copyright

5 A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

Priority

10 This application claims priority benefit to U.S. provisional patent application Serial No. 60/188,546 filed March 10, 2000 and entitled "Memory Interface and Method of Interfacing Between Integrated Circuits."

Related Applications

15 This application is related to pending U.S. patent application Serial No. 09/418,663 filed October 14, 1999 entitled "Method and Apparatus for Managing the Configuration and Functionality of a Semi-Conductor Design", which claims priority benefit of U.S. provisional patent application Serial No. 60/104,271 filed October 14, 1998, of the same title.

20

Background of the Invention

1. Field of the Invention

25 The present invention relates to the field of integrated circuit design, specifically to the integration of peripheral components and macro functions with a central processing unit (CPU) or user-customizable microprocessor.

2. Description of Related Technology

As semiconductor processing capabilities increase the number of transistors that can be economically built on a single Integrated Circuit (IC), systems designers

are made less effective by the difficulty encountered in combining large-scale macro blocks on a single IC. Such large-scale macro blocks (or “macro functions”) include, for example, those associated with third generation (“3G”) communications architectures, such as functions performing Viterbi butterfly decode, cyclic redundancy checks (CRC), convolutional encoding/decoding, permutation, and carrier modulation/demodulation. Some of the problems encountered by the designer are underscored by the need to integrate special purpose functions with an existing instruction set implemented by a given central processing unit (CPU). Often, a non-integrated design approach is employed, wherein the large-scale macro blocks or functions are treated as separate entities from the processor core, thereby requiring additional complexity, as well as specialized or unique interfaces between the core and its associated functions which are not standardized across the device. Specifically, with respect to memory interfaces, the use of control registers associated with the memory ports of the interface not only complicates the design, but also may under certain circumstances limit or restrict the functionality of the interface. For example, individual macro blocks associated with the design may be precluded from acting on data in separate memory banks simultaneously, thereby hindering the performance of the design as a whole by requiring that memory accesses be performed in “lock-step” fashion.

Prior art treatment of large-scale macro functions as separate entities within the design has further disabilities relating to memory. In particular, since the macro block is effectively a separate entity from the core, memory interfaces to existing core memory are often quite complex, thereby often necessitating the provision of separate memory dedicated to the macro function (or shared between multiple macro functions). The requirement for such additional memory adds cost and complexity to the device, as well as monopolizing already precious real estate on the die. This is especially true for so-called “system-on-a-chip” (SoC) designs, where available memory is often a limiting parameter. Additionally, such dedicated “off-core” memory is by definition not local to the core, and hence results in increased latency when such memory must be accessed by the core.

Furthermore, as more such large-scale macro function blocks are added to the design, the propensity for such increased complexity and non-standardization across the design increases accordingly.

Furthermore, conventional interface mechanisms are typically based on a common bus, and transfers between peripherals and the core(s) are arbitrated by one or more direct memory access (DMA) controllers. However, under such an approach, the timed transfer of data may not be deterministic, which is often a crucial requirement for DSP applications. Specifically, DSP systems often require not only that data are processed correctly mathematically, but that results are delivered at the right time. In this sense a "deterministic" transfer is one for which the timing is exactly known.

Based on the foregoing, there is a need for an improved apparatus and method for enabling macro functions and peripherals present on an integrated device to interface with the device processor core in a simple and standardized manner. Such improved interface would not only allow for standardized interface between macro-functions across the device, but also allow multiple macros to interface with individual (or a plurality of) memory banks simultaneously. Such improved apparatus and method would also ideally obviate separate or discrete local memory now used in support of macro (e.g., DSP core) functions, and facilitate deterministic transfer of data between functional entities in the design.

Summary of the Invention

The present invention satisfies the aforementioned needs by an improved apparatus and method for interfacing between integrated circuit devices, such as between a digital processor and associated memory array.

In a first aspect of the invention, an improved interface apparatus useful for interfacing between an external device and one or more processor functions is disclosed. In one exemplary embodiment, the external device comprises a memory array, and the interface comprises a plurality of memory ports, a plurality of function ports with associated function controllers, a crossbar connecting the memory ports to the function ports, and an arbitration unit for arbitrating memory accesses and facilitating burst load/store operations. The interface apparatus enables a macro block

to be integrated with the CPU, controlled with a custom processor instruction, and act directly on CPU local memory banks. Other macro blocks can also access data in other banks simultaneously. Under CPU control, blocks of data can be acted on by macro functions in a pipelined fashion. The local CPU memory banks therefore replace
5 memory that might otherwise be duplicated in the macro block. Once acquired, data is local to the CPU which can also act on it without unnecessary additional latency.

In a second aspect of the invention, an improved data transfer protocol used in conjunction with the aforementioned interface is disclosed. In one embodiment, the protocol comprises a synchronous protocol using a plurality of registers disposed
10 within the aforementioned functional control units of the function ports. Data exchange with the memory interface is synchronized with the clock of the processor core, and data is strobed in and out of the memory after memory access has been requested and granted. The grant of memory access requests is delayed if the requested memory bank is being accessed for a burst load/store operation, or by the processor
15 core. The protocol of the invention supports, in conjunction with the memory interface, multiple simultaneous strobed accesses to different memory banks by different macro functions (function ports) associated with the interface.

In a third aspect of the invention, an improved data processing apparatus is disclosed. The device generally comprises a processor core, the aforementioned
20 memory interface, at least one macro function, XY memory array, and I/O interface. In one exemplary embodiment, the processor core comprises an extensible RISC-based digital processor, and the macro function comprises a digital signal processor (DSP). The DSP may be a general purpose DSP, or alternatively any one of a number of algorithmically optimized designs which are adapted to perform certain data
25 processing tasks. The RISC processor and DSP are tightly coupled such that the DSP and memory interface effectively become part of the RISC processor's instruction set, the macro function (DSP) being controlled by, for example, decoded instructions generated by the pipeline decode stage of the RISC processor. Furthermore, peripheral devices can have direct memory access (DMA) capability with respect to the XY
30 memory array via the I/O interface. The components are also advantageously combined into a single-die integrated circuit device. In another embodiment, the

device comprises a "3G" ASIC having a plurality of macro blocks including a signal receiver and demodulator, "turbo" or Viterbi decoder, block cyclic redundancy code (CRC) calculation macro, block permute macro, block convolution encoder macro, and modulator and transmitter, all of which are coupled to the core memory array via the memory interface.

In a fourth aspect of the invention, a method of interfacing between a memory and one or more processor functions is disclosed. In one exemplary embodiment, the method comprises specifying a number of function ports for an interface; specifying a number of memory ports for the interface; setting a number of control and other registers based on the function port configuration; specifying an interface protocol; initiating the function(s) associated with the function port(s); arbitrating one or more memory accesses; and performing read/write access to memory according to the selected protocol and control by the parent processor core.

In a fifth aspect of the invention, a method of testing a function associated with the previously described memory interface is disclosed. In one exemplary embodiment, the method comprises providing a memory interface having at least one memory port and associated memory array, at least one function port and associated function, and at least one control register used for controlling said function; providing a test sequence; providing an input test value in the memory array; initiating the function; generating results from the operation of the function on the input test value; and comparing the results returned by the function against a known value in order to test the operation of the function.

In a sixth aspect of the invention, an improved method of designing an integrated circuit device having an extensible processor core, secondary processor (e.g., DSP) or macro function, and memory interface is disclosed. In one embodiment, the method comprises providing an extensible core; providing at least one macro function; providing at least one memory interface; adding an HDL "wrapper" around the DSP or macro function, the HDL wrapper adapted to (i) translate signals, (ii) buffer memory interfaces, and (iii) synchronize clock signals with the memory interface. In another embodiment, the method comprises providing an extensible core; providing at least one "soft" macro function; providing at least one memory interface

as described previously herein; and adapting the “soft” macro function implementation to meet the specification associated with the memory interface.

Brief Description Of The Drawings

5

Fig. 1 is a plot of data sample rate versus type of application, illustrating the relationship between various types of processor architectures and their possible applications.

10

Fig. 2 is a block diagram of one exemplary embodiment of the memory interface according to the present invention.

Fig. 2a is a logical block diagram of one exemplary embodiment of logic adapted to provide decoded instruction and operand information from the parent processor's decode pipeline stage to the memory interface of the invention.

15

Fig. 3 is a block diagram illustrating a first application of the interface of Fig. 2, wherein the DSP core is "tightly" coupled to the parent processor core (e.g. ARC) and XY memory.

Fig. 4 is a block diagram illustrating a second application of the interface of Fig. 2, wherein the DSP core is served by a separate I/O interface, the DSP core being controlled by the processor core.

20

Fig. 5 is a block diagram illustrating a third application of the interface of Fig. 2, wherein the interface is used to interface directly between an I/O device and the XY memory.

Fig. 6 is block diagram of one embodiment of the interface of the invention, illustrating the various signals and registers associated therewith.

25

Fig. 6a is a logical flow diagram illustrating one exemplary embodiment of the method for testing a macro function using the memory interface of the present invention.

Fig. 7 is a timing diagram illustrating one embodiment of the protocol used in conjunction with the interface of the present invention.

30

Fig. 8 is a block diagram of one exemplary embodiment of an integrated processor device including a processor core, DSP core, XY memory, and the memory interface of the present invention.

Fig. 9 is a block diagram of a second embodiment of an integrated processor device including the memory interface of the present invention and a plurality of macro function entities, the processor device and macro function entities being adapted for 3G communications.

5 Fig. 9a is a logical block diagram illustrating the operation of one exemplary embodiment of the macro function “pipeline” using the memory interface of the present invention.

Fig. 10 is a logical flow diagram illustrating one embodiment of the method of interfacing a function with a memory array according to the invention.

10 Fig. 11 is a logical flow diagram illustrating one exemplary embodiment of the method of generating a design for an integrated circuit device having a parent processor, a memory interface, and at least one macro function associated therewith, wherein an HDL “wrapper” is used as the macro function interface.

15 Fig. 11a is a logical flow diagram illustrating one embodiment of the method of adding an HDL wrapper according to Fig. 11.

Fig. 12 is a logical flow diagram illustrating a second embodiment of the method of generating a design for an integrated circuit device having a parent processor, a memory interface, and at least one macro function associated therewith, wherein a “soft” macro function is utilized and adapted to the requirements of the memory interface.

20

Detailed Description

Reference is now made to the drawings wherein like numerals refer to like parts throughout.

25 As used herein, the term “processor” is meant to include any integrated circuit or other electronic device capable of performing an operation on at least one instruction word including, without limitation, extensible reduced instruction set core (RISC) processors such as the ARC™ user-configurable core manufactured by the Assignee hereof, central processing units (CPUs), and digital signal processors (DSPs). Furthermore, various functional aspects of the processor may be implemented
30 solely as software or firmware associated with the processor.

As used herein, the term "parent" processor refers generally to the aforementioned ARC core (or similar), while the term "host" processor refers generally to an external processor which controls the operation of the ARC core and/or other functional aspects of the design.

5 Additionally, it will be recognized by those of ordinary skill in the art that the term "stage" as used herein refers to various successive stages within a pipelined processor; i.e., stage 1 refers to the first pipelined stage, stage 2 to the second pipelined stage, and so forth.

10 It is also noted that while portions of the following description are cast in terms of VHSIC hardware description language (VHDL), other hardware description languages such as Verilog® may be used to describe various embodiments of the invention with equal success. Furthermore, while an exemplary Synopsys® synthesis engine such as the Design Compiler 2000.05 (DC00) is used to synthesize the various embodiments set forth herein, other synthesis engines such as Buildgates® available
15 from, *inter alia*, Cadence Design Systems, Inc., may be used. IEEE std. 1076.3-1997, IEEE Standard VHDL Synthesis Packages, describe an industry-accepted language for specifying a Hardware Definition Language-based design and the synthesis capabilities that may be expected to be available to one of ordinary skill in the art.

20 Overview

 The memory interface of the present invention has been conceived to enable, *inter alia*, DSP macro functions and peripherals to interface with another processor core using a simple and standard methodology. Close integration with predefined VLSI functions increases the ability to satisfy demanding applications and meet
25 emerging industry standards, such as those relating to so-called "3G" applications. For certain high-speed communications tasks, only custom DSP and input/output (I/O) functions can meet the processing demands. This concept is illustrated in Fig. 1.

 The extensible nature of certain processor cores (e.g., the Applicant's "ARC" core) and associated XY memory allow DSP and I/O functions to be tightly coupled
30 for such demanding applications. Using the apparatus and methodology of the present

invention, the DSP core(s) effectively become part of the parent processor core instruction set, and I/O peripherals have direct memory access (DMA) to the processor core.

Furthermore, many algorithmically optimized DSP core designs exist. Combining dedicated hardware functional performance and software flexibility within the same IC advantageously provides the most cost effective and shortest time to market for new product development.

Description of Interface

Referring now to Fig. 2, one exemplary embodiment of the memory interface of the present invention is described. As shown in Fig. 2, the interface 200 comprises generally a plurality of memory ports 202, a plurality of function ports 204, a plurality of function controllers 206 associated with the aforementioned function ports 204, and an arbitration controller 208. The interface 200 is designed to interface between various “macro” functions 210 associated with a logic or processing device typically in the form of an integrated circuit (IC) such as a DSP, microprocessor, or ASIC (hereinafter generally referred to as “IC function”), and a memory array 212 having a plurality of individual memory banks 214. While an interface 200 having a plurality of ports 202, 204 is shown, it will be recognized that the interface device 200 of the present invention may be implemented with any lesser number of ports, such as one memory port 202 and one function port 204.

The number of function ports 204 (and hence macro functions 210) is determined by the algorithmic needs of a particular application, and the necessity for hardware acceleration in that application. The number of memory ports 202 (and hence memory banks 214) is determined by the virtual flow of data between macro function blocks, the latter equivalent to a macro function block processing pipeline under CPU control. If a macro block 210 is processing data from a peripheral device (such as shown in Fig. 6 herein) before storing to CPU local memory, then the number of memory banks 214 is determined by the buffering requirements associated with the macro block function, and the ability of the CPU to process data in software.

The memory ports 202 of the apparatus of Fig. 2 comprise interfaces with the banks 214 of the array 212. Advantageously, there are no control registers associated with the memory ports 202; rather, control is performed via the associated IC function 210. The interface 200 arbitrates access to each of the memory banks 214 using the arbitration controller 208. The memory ports of the illustrated embodiment comprise simple address, data, read, write, select, and control signals required by typical random access memory (RAM) design instantiations as are well known in the semiconductor arts, thereby making optimal use of existing technology.

One embodiment of the arbitration controller logic is described. The arbitration controller 208 comprises a multiplexer adapted to select between burst, direct memory interface (DMI) devices, debug (not shown) functions, and the processor. The construction and operation of multiplexer devices are generally well known in the semiconductor arts, and accordingly not described further herein. The multiplexer of the present invention, however, is controlled by logic which provides the following priority structure: (i) the burst address is selected with the highest priority; (ii) external device permission to access the memory of the bank is given next priority; (iii) the debug channel is selected when the parent processor core (e.g., ARC) is halted, and the host device attempts to access the XY memory; and (iv) the processor source and destination operand busses are otherwise selected if appropriate. It will be recognized that other priority structures may be implemented consistent with the invention, however.

The function ports 204 comprise the interface of the memory interface 200 with the IC functions 210. All data, control, and clock signals are routed through the function ports 204. A synchronous protocol, described in detail herein with respect to Fig. 7, is used in the present embodiment to facilitate read/write data transfer through the ports 204, although it can be appreciated that other protocols (synchronous or non-synchronous) may be substituted. It will be recognized that the illustrated arrangement shows a minimum interface requirement, and other more complex or functionally enhanced arrangements (or combinations thereof) can be substituted. For example a virtual component interface (VCI) of the type well known in the art could be used consistent with the invention.

The function controllers 206 comprises the control, status, and test registers (see Tables 4-6 below) associated with each of the IC functions 210. The function controllers also include an interface (as illustrated in the exemplary configuration of Fig. 2a) to the core processor's stage two instruction decode and source operand value.

5 The interface 200 of the invention further utilizes a data transfer "fabric" which interconnects the memory ports 202 with the function ports 204, thereby facilitating data distribution within the interface. In the embodiment illustrated in Fig. 2, the fabric comprises a crossbar arrangement (represented by the series of arrows 216 of Fig. 2) for data communication between the ports 202, 204, although it will be appreciated
10 that other techniques for (selectively) communicating data from one or more ports to one or more ports may be utilized. The construction and operation of crossbar switches is well known in the electronic arts, and accordingly is not described further herein.

Using the aforementioned crossbar arrangement, each IC function 210 is connected through the interface 200 to a bank 214 of the memory array 212, and
15 multiple functions/banks may be connected simultaneously. In the illustrated embodiment, the memory array 212 comprises XY memory. Such memory may comprise, for example, static random access memory (SRAM), dynamic random access memory (DRAM), synchronous DRAM (SDRAM), double-data rate (DDR) SDRAM, embedded/flash memory, or any other type of memory suitable for the
20 desired application. Such memory devices are well known in the semiconductor arts, and accordingly will not be described further herein. The determination of which memory bank 214 in the array 212 is accessed is made either by (i) a control register in processor core auxiliary register space; or (ii) the instruction operand (e.g. "FFT 2" of the aforementioned ARC core). For example, in the case where the IC function
25 comprises a DSP, the individual functions associated with the DSP can access any of the XY memory banks 214 in the array 212 for intermediate calculations and results. In one embodiment of the invention, the XY memory is implemented as two pages of memory, and adapted such that two operand sources may be provided simultaneously. Alternatively, provision may be made for selection between one of the two pages of
30 memory at any one time if desired.

It is noted that different DSP cores (or other types of processor cores) will generally have varying interface, control and memory requirements. The interface 200 of the present invention advantageously provides the designer with the ability to integrate cores of various configurations directly with the parent processor core (e.g., extensible ARC processor) instruction and operand decode mechanism, auxiliary register, and on-core memory resources. The chip/system designer may choose to either to add an HDL "wrapper" around the DSP or core to translate signals, buffer memory interfaces, and synchronize clock signals, or alternatively modify the "soft" DSP core implementation to meet the specification associated with the memory interface 200, as described below with respect to Figs. 11a and 11b.

Referring now to Figs. 3-5, exemplary applications of the memory interface 200 of the present invention are described, illustrating how the interface 200 can be used to integrate devices such as DSP cores, ASICS, or other types of peripherals. As illustrated in Fig. 3, a DSP core 302 may be tightly coupled to an associated processor core (e.g., ARC) using the interface 200. As used herein, the term "tightly coupled" refers generally to the degree of interaction and coherence between the DSP core, interface, and parent processor. In the example of Fig. 3, the DSP core 302 can act on buffered data contained in the XY memory bank 212. The interface 200 of the present invention allows the DSP core 302 to be initiated by an instruction from the parent processor core (not shown), and uses the parent processor core extension registers for control. Specifically, auxiliary registers may be mapped to control the macro function block 210 and report status, and may be accessed through "LR" and "SR" instructions, as illustrated in the following example:

```

ld %r0, #CONTROL_WORD
sr %r0, [macro_control] ; load the control word into auxiliary reg.

ld %r0, #BANK
go_macro 0, %r0 ; source operand contains bank to act on

:
:
Irq_macro_ready:
lr %r0, [macro_status] ; read status

```

As illustrated in Fig. 4, some DSP core functions (such as calculations performed in support of the aforementioned high-speed 3G communications) may require data at a higher rate than can be provided by the parent processor core. In the example of Fig. 4, the DSP core 302 is served with data by a local peripheral device 404 (such as a data storage device), and delivers "processed" results (such as, for example, the output of a FFT algorithm, packing/unpacking data from a high data rate bit stream, or Viterbi decode metrics) directly into the XY memory 214 of the processor. The interface 200 further provides support for interrupts and flags to indicate, *inter alia*, block filled or empty status. As shown in Fig. 6, DSP function status signals may be connected to bits in registers mapped into the parent processor's auxiliary register space. Similarly, bits in that register may be used to enable the routing of interrupt signals from the DSP function to processor's interrupt controller.

As illustrated in the example of Fig. 5, the I/O peripheral device 404 may also interface directly with the memory interface 200. In this manner, unprocessed or "raw" data may be delivered directly to the various memory banks 214 within the memory array 212 as arbitrated by the interface 200. Hence, the memory interface advantageously allows the memory array to act as a multi-bank buffer for storage and subsequent read-out of such raw data.

In terms of operation, the functions of the DSP core 302 (or other IC function in communication with the interface 200) may, if desired, be initiated by the processor core (e.g. ARC™) pipeline stage 2 instruction decode, and also may optionally use the operand value. Other schemes of DSP core or peripheral initiation may also be used consistent with the invention. For example, it is also possible to initiate and control the DSP core or IC function using one or more control/status registers within the auxiliary space associated with the parent processor.

It is also noted that one or more XY memory banks 214 within the array 212 can advantageously be filled by a "burst" mode load while DSP functions execute on other memory banks. The memory interface 200 and the X/Y memory burst control unit (not shown) arbitrate between accesses. The burst control unit provides direct

memory access (DMA) between the main memory and the XY memory system. Its functionality includes the ability to transfer blocks of data, and in one embodiment is integrated with the existing XY system. Note also that the arbitration unit of the present invention includes logic to arbitrate between the processor, an external DSP core (via DMI), host debug port (also not shown) and the parent processor, as previously described.

In one embodiment, the occupied memory resource is "locked out" by one access to another using a first-in-time method (i.e., whichever access is initiated first will lock the other access out until completed), as illustrated below:

```

ld %r0, #CONTROL_WORD
sr %r0, [macro_control] ; load the control word into auxiliary reg.
go_macro 0, %r0 ; source operand contains bank to act on
mov %r0, x0_u

```

Copyright ©2001 ARC International plc. All rights reserved.

In the foregoing example, the move (mov) instruction is attempting to transfer data from XY memory into a core register. Two arbitration solutions are possible: (i) either stall the move operation until the macro function completes, or (ii) with additional control logic, stall the macro function and allow processor access to a non-conflicting memory region.

It will be recognized, however, that other types of memory arbitration of the type well known in the art may be utilized to provide non-conflicting memory resource access, round-robin or prioritized arbitration with time division, queued or packetized transfer.

If either the aforementioned burst mode or DSP core function have control over X/Y memory, then "read or "write" operations by the parent processor core to that memory bank will be invalid, but this will not stall the processor core, as block transfers are, in the present embodiment, permitted to take multiple cycles. Hence, in this embodiment, the programmer has the responsibility of ensuring that such

automated memory accesses, once initiated, have completed, although other mechanisms for ensuring memory access operation completion may conceivably be used. The following shows a polled solution to invalid parent processor XY access during burst mode:

5

```
sr          [burst_control], #INITIATE_BURST      ; DMA data into
XY memory.
```

```
; No access to same XY
```

```
bank here
```

10

```
Wait:
```

```
lr          %r0, [burst_status]
```

```
bne        Wait
```

```
; Wait for DMA to
```

```
complete
```

15

```
mov        %r0, x0_u
```

```
; XY access allowed
```

```
Copyright ©2001 ARC International plc. All rights reserved.
```

20

Data Transfer Protocol

25

When the DSP core 302 or I/O device 404 (Figs. 3-5) requests access to a memory bank 214 within the array 212, access is arbitrated by the interface 200, specifically the arbitration controller 208 (Fig. 2). The interface 200 of the invention provides data, address, XY page and read/write selection using a memory request/memory grant system which is synchronous with the system clock (CLK). Fig. 6 illustrates a simple interface configuration 600, based on a standard bus-request/bus-grant mechanism, having only one memory port 602 and one function port 604. The interface 600 is coupled via the function port 604 to a DSP function 610, which is

30 coupled to an I/O peripheral 620. The X/Y memory banks 614 interface directly with the memory port 602 of the interface 600. The function controller 606 includes the data, control, and debug register set for the function port 604, and also interfaces with the stage 2 instruction decode (1-bit) and operand (32-bit) of the parent processor core.

35

The signal set associated with the exemplary interface 600 of Fig. 6 is now described with respect to Tables 1-3 below. Table 1 lists interface signals for the interface 600 and their associated functions. Table 2 lists memory request/grant

signals. Table 3 lists register control signals and signals generated by the processor core (e.g. ARC™) used to initiate function execution and to provide immediate operands from pipeline stage 2, as previously described.

In the present invention, the control signals used to initiate the macro function are duplicated in one or more control registers. Specifically, in one embodiment, the signals are connected such that a “write” operation to the control register in auxiliary space is registered as a valid command as if correct in stage 2 of the processor pipeline. This ability, *inter alia*, facilitates the testing of the macro block through the host interface, as described below in greater detail. As auxiliary registers will be accessible from the host interface, this design feature advantageously ensures that the function can be tested. For example, a test sequence might write an input test vector into XY memory, initiate the macro function, and then compare returned results against a known vector. This process is depicted graphically in Fig. 6A. Such process may be applied to any number of different operations including, for example, decoding of convolution-encoded data by a Viterbi function (and “Turbo” decode), or correct calculation of a CRC for a block of data.

Additional auxiliary registers may be used for test purposes as well. For example, in Applicant's ARC core, four test registers are configured for each function port by default. Specifically, if a macro block is connected to a peripheral device, the four test registers can be used to simulate the action of that peripheral under software control. Comparison of transform results in memory to the known stimulus can therefore be achieved for validation of the correct functioning of the macro block 210, advantageously without having to generate real-world stimuli. Other uses of the test registers are possible depending on the testing/operational requirements. It will also be recognized, however, that such additional auxiliary registers are optional, and may be specified in any number desired by the programmer/designer as allowed by the hardware constraints.

In the present embodiment, a number of read/write registers associated with each function port are provided. A default value is normally set, but as with the auxiliary registers, this number may vary. These read/write registers are used for

control and data requirements specific to the function associated with the function port, such as, for example, error reporting such as internal data saturation of a fast fixed-point FFT block, status of a connected peripheral, etc.

One or more control registers are also provided. A basic requirement of the control register in the present invention is that power control and external function reset capabilities are provided. This reset (or other control) forces the function to release the memory bank back to the parent processor core. Interrupt control is provided and enabled by the control register, but the IC function also reports internal function status to the processor core using flags.

Tables 4-6 list registers used within the exemplary interface 600 of Fig. 6. Table 4 lists control/status registers; in the illustrated embodiment, two control/status registers are provided which control read/write operations. Table 5 lists general purpose registers; four general purpose registers are provided within the interface 600 for passing control parameters and data to the IC function, and for implementing special features. For example, the control parameters passed to the IC function could comprise FFT size or window type, and the IC function could return a block exponent or cyclic redundancy code (CRC) via the general purpose registers. Another example is for a DES instruction having the key code and accumulators as extension registers. Many other uses are possible.

Table 6 lists test/debug registers within the interface 600; four registers are provided in the present embodiment for, *inter alia*, function specific testing and debug capability. For example, as previously described, if a macro block is connected to a peripheral device the test registers might be used to simulate the action of that peripheral under software control, with comparison of transform results in memory to the known stimulus without having to generate real-world stimuli.

Table 1. Interface Signals

Signal	Description
DMI_DATA	Data read/write bus from custom function. 16/32-bit data.
DMI_ADDR	Address bus from custom function. Bus size depends on bank size.
DMI_X/Y	Signal from custom function selects X or Y bank for read or write.
DMI_1632	Signal from custom function selects 16 or 32-bit addressing and data bus mode.
DMI_R*/W	Signal from custom function requests read or write.
DMI_BNK	Signal from custom function requests XY bank for transfer.

Table 2. Memory Request/Grant Signals

Signal	Description
DMI_CLK	System clock output from ARC Memory Interface.
DMI_MR	Memory request signal from custom function.
DMI_MG	Memory grant signal from ARC Memory Interface.
DMI_DS	Data strobe signal from custom function.

Table 3. Register Control Signals

Signal	Description
DMI_CTRL	Control signals output from ARC Memory Interface control/status register.
DMI_STAT	Status signals from custom function.
DMI_IRQ	Interrupt signals from custom function (enabled in control/status register).
DMI_INS	ARC state 2 instruction decode (via ARC Memory Interface) used to initiate function execution.
DMI_OP	ARC state 2 operand (via ARC Memory Interface) used for basic function parameters.

Table 4. Register Control Signals

Signal	Description
DMI_CTRLx	Write: IRQ enable, reset, power, run/stop, clear error, test mode, free bits
DMI_Opx	Write: Set operand for function initiation via DMI_CTRLx. Read: An error code may be returned by a read.

Notes:

1. Default bit positions are defined in the implementation specification.

5 For example, the settings of Table 4a are representative bit positions:

Table 4a – Representative Bit Positions

8	7	6	5	4	3	2	1	0
RESET	IRQ_ENABLE	POWER	RUN	CLEAR_ERROR	TEST_MODE3	TEST_MODE2	TEST_MODE1	TEST_MODE0

2. An "x", in the register name specifies the function to which the register applies. All registers of the illustrated embodiment are 32-bits.

Table 5. General Purpose Registers

Signal	Description
DMI_GPOx	Function specific read/write register.
DMI_GP1x	Function specific read/write register.
DMI_GP2x	Function specific read/write register.
DMI_GP3x	Function specific read/write register.

Table 6. Test/debug Registers

Signal	Description
DMI_DBG0x	Function specific read/write test/debug register.
DMI_DBG0x	Function specific read/write test/debug register.
DMI_DBG0x	Function specific read/write test/debug register.
DMI_DBG0x	Function specific read/write test/debug register.

It is noted that while the foregoing embodiment described a specific number of signals and registers associated with these signals, variations in the configuration of the interface, including the number and function of signals, and/or the number of registers, may be employed depending on the specific application and needs of the designer/programmer.

Referring now to Fig. 7, one embodiment of the interface protocol according to the invention is described. While the DSP function 610 may have its own clock, data exchange with the interface 600 is, in the present embodiment, synchronized with the clock of the processor core. Data is strobed in and out of the memory 612 after memory access has been requested and granted as previously described.

As illustrated in Fig. 7, the clock signal (DMI_CLK) 701 comprises a regular periodic clock signal of the type well known in the art. During a write cycle 702, the memory request and grant signals (DMI_MR, DMI_MG) 703, data strobe signal (DMI_DS) 704, data read/write signal from the IC function (DMI_R/*W) 706, and addressing/bus size select/X or Y memory bank select signals (DMI_ADDR, DMI_X/Y, and DMI_1632, respectively) 708, 710, 712 are set as indicated in Fig. 7 to perform a write operation from the data bus (DMI_Data) 714 to the selected address within the X or Y bank via the interface 600. Conversely, during the read cycle, the interface 600 sets the aforementioned signals 703, 704, 706, 08, 710, 712 as appropriate to load data from the selected memory bank 614 and transfer it to the IC function using the interface 600; i.e., via the memory port, fabric, and function port to the IC function.

Fig. 8 illustrates an exemplary pipelined processor (system) fabricated using a 1.0 micron process. As shown in Fig. 8, the processor 800 includes, *inter alia*, a processor core 802, on-chip read-only memory 804, XY random access memory 806, a DSP core 808, memory interface 200, ADC 812, DAC 814, custom analog and/or digital circuitry 816, and an external interface 818. The device is fabricated using the customized VHDL design methodology of Applicant's co-pending U.S. patent application Serial No. 09/418,663 entitled "Method and Apparatus for Managing the Configuration and Functionality of a Semiconductor Design" filed October 14, 1999,

which is incorporated herein by reference in its entirety. The interface 200 of the present invention may advantageously be integrated directly into the configuration environment described therein, as discussed in greater detail below with respect to Figs. 11-12. Many of the interface 200 configuration parameters, such as the number of memory ports, may be inherited directly from the XY memory configuration specified in this environment. The generated design is subsequently synthesized into a logic level representation, and then reduced to a physical device using compilation, layout and fabrication techniques well known in the semiconductor arts.

It will be appreciated by one skilled in the art that the processor of Fig. 8 may contain any commonly available peripheral such as serial communications devices, parallel ports, timers, counters, high current drivers, LCD drivers, memories and other similar devices. The present invention is not limited to the type, number or complexity of peripherals and other circuitry that may be combined using the method and apparatus. Rather, any limitations are imposed by the physical capacity of the extant semiconductor processes which improve over time. Therefore it is anticipated that the complexity and degree of integration possible employing the present invention will further increase as semiconductor processes improve. For example, the present invention is compatible with 0.35, 0.18, and 0.1 micron processes, and ultimately may be applied to processes of even smaller or other resolution. An exemplary process for fabrication of the device is the 0.1 micron "Blue Logic" Cu-11 process offered by International Business Machines Corporation, although others may be used.

Fig. 9 illustrates yet another embodiment of an integrated circuit 900 fabricated using the apparatus and methods of the present invention. Specifically, the IC comprises an application specific integrated circuit (ASIC) embodying a "3G" (i.e., third generation) communications application having a plurality of macro functional blocks 210. The macro functional blocks 210 of the memory interface 902 include a signal receiver and demodulation block 904, "turbo" or Viterbi decoder block 906, block CRC calculation macro block 908, block permute macro block 910, block convolution encoder macro block 912, and modulation and transmit block 914. The memory banks 920 act to form circular buffers of the type well known in the data

processing arts. Data remains in the memory banks 920 and is acted on by the macro blocks 210 in sequence.

Furthermore, multiple macro blocks may be active as a “macro pipeline” controlled by the CPU as illustrated in Fig. 9a. Specifically, the macro functions, under CPU control via customized instructions within the base or extension instruction sets of the parent processor, sequentially act on data transferred to or out of the memory array in lockstep or pipelined fashion via the memory interface of the invention. Fig. 9a illustrates the pipelined flow of block data in a 4-memory bank, 4-function configuration of the present invention. Exemplary pseudo-code for this operation is as follows:

```

for(;;)
{
    function0(bank[(cycle+0)%BANKS]); /* Initiate functions */
    function1(bank[(cycle+1)%BANKS]);
    function2(bank[(cycle+2)%BANKS]);
    function3(bank[(cycle+3)%BANKS]);

    while(all_functions_not_complete); /* Wait */
    cycle++;
}

```

Copyright ©2001 ARC International plc. All rights reserved.

Referring now to Fig. 10, the method of interfacing a function with a memory array according to the invention is described. In the first step 1002 of the method 1000, a number of function ports and memory ports are specified or defined for the interface. As previously described, the interface may, in one example, inherit the number of memory ports from the parent processor core design (as specified by the user/design constraints).

Next, in step 1004, the number of control and other registers needed for the interface is set based on the function port configuration. In one exemplary embodiment, a library of macro blocks with compatible interfaces is instantiated in the design by the (e.g. the semiconductor synthesis and design software manufactured by the Assignee hereof, and described in detail in Assignee’s co-pending U.S. patent application Serial No. 09/418,663, previously incorporated herein). Under such

approach, the aforementioned software has knowledge of the block's requirements, and can instantiate the interface block appropriately.

The interface protocol is then specified in step 1006. One exemplary synchronous protocol is described herein with reference to Fig. 7. In step 1008, the function(s) associated with the function port(s) is/are initiated. As previously described, the functions may be initiated by the processor core stage 2 pipeline decode, using one or more control/status registers in auxiliary memory space, or even other methods. In step 1010, memory accesses are arbitrated by the crossbar/arbitration unit, 216/208 (Fig. 2) based on the memory access request/grant scheme previously described, or other arbitration scheme. Lastly, in step 1012, the read/write access to memory are conducted according to the selected protocol and control by the parent via one or more extension registers.

Referring now to Fig. 11, an improved method of designing an integrated circuit device having an extensible processor core, secondary processor (e.g., DSP) or macro function, and memory interface is described. As shown in Fig. 11, one embodiment of the method 1100 generally comprises first providing an HDL representation of an extensible core cell, such as the ARC user-configurable core previously described herein (step 1102). Next, HDL representations of one or more macro functions (blocks) are provided per step 1104. Examples of macro functions include the Viterbi decode, convolutional encoding, or CRC block previously described. Next, in step 1106, HDL representations of one or more memory interfaces as described herein with respect to Fig. 3 are provided. An HDL "wrapper" (i.e., a complementary HDL description designed to adapt to, and interface with, the selected macro functions) is then generated and disposed "around" the aforementioned macro function (s) in step 1108. As used herein, the terms "wrapper" and "around" are not necessarily used in a physical or spatial context, but rather in a figurative sense in that the HDL wrapper functionally envelopes the macro function block(s) (e.g., DSP core) so as to provide proper communication between the existing function block and the memory interface of the invention. Specifically, such communication includes (i) translating signals transmitted to and from the memory interface, (ii) buffering memory operations, and (iii) synchronizing clock signals of the macro function with the memory interface. Note that in the embodiment of

Fig. 11, the pre-defined functions and protocols associated with the macro function block(s) (e.g., DSP) are preserved; the HDL wrapper acts in effect as a translation and interface medium between the macro function and memory interface device, the latter ultimately coupled to the memory array.

5 Fig. 11a illustrates one embodiment of the method of designing an integrated circuit adding the HDL wrapper according to step 1108 of the method 1100 of Fig. 11. In general, the system builds from the master database a hierarchical directory structure containing HDL files that fully describe the parent processor. The memory interface is included in the master database and, once the memory interface and other related
10 parameters are selected by the user, are included in the directory structure.

The memory configuration (i.e., use of memory interface and number of memory ports and function ports) is first specified by the user (step 1120). The system builder script is then invoked (step 1122) to (i) create a working directory of the user files, and (ii) copy files, including VHDL for macro functions and memory interface as required
15 from the master database. The selected files are also customized as required to configure the system as selected by the user. The structural VHDL is then generated per step 1124. A simulation makefile and a synthesis script are then generated (step 1126). The designer then simulates or synthesizes the design (including memory interface and macro functions) per steps 1128 and 1130, respectively.

20 Fig. 12 illustrates a second embodiment of the method of designing an integrated circuit according to the invention. The method 1200 generally comprises first providing an extensible core description as previously described (step 1202). Next, at least one “soft” macro function is provided per step 1204. As used herein, the term “soft” refers to a macro function which may be selectively configured by the designer. Such soft macro
25 functions effectively have the HDL wrapper previously described with respect to step 1108 of Figs. 11 and 11a above incorporated directly into their design. The wrapper furthermore may be made configurable itself.

The memory interface description is next provided per step 1206. In step 1208, the parent processor parameters such as cache size, extension instructions, and type of
30 build (e.g., system versus core only) is selected. The memory configuration (i.e., use of memory interface and number of memory ports and function ports) is also specified. The

system builder script is then invoked (step 1210) to (i) create a working directory of the user files, and (ii) copy files, including VHDL for modules and extensions, as required from the master database. One of the copied files in the present example comprises the memory interface file, while another comprises that for the selected “soft” macro functions previously identified for the build by the user. The selected files are also customized to configure the system as selected by the user. The structural VHDL is then generated per step 1212. A simulation makefile and a synthesis script are then generated (step 1214). The designer then simulates or synthesizes the design (including extended core, memory interface, and user-configured macro functions) per steps 1216 and 1218, respectively. Note that in contrast to the method of Fig. 11, the method of Fig. 12 integrates the memory interface with the “soft” macro function(s) (and processor core) during design generation, thereby modifying the macro function configuration.

It will be recognized that while certain aspects of the invention are described in terms of a specific sequence of steps of a method, these descriptions are only illustrative of the broader methods of the invention, and may be modified as required by the particular application. Certain steps may be rendered unnecessary or optional under certain circumstances. Additionally, certain steps or functionality may be added to the disclosed embodiments, or the order of performance of two or more steps permuted. All such variations are considered to be encompassed within the invention disclosed and claimed herein.

While the above detailed description has shown, described, and pointed out novel features of the invention as applied to various embodiments, it will be understood that various omissions, substitutions, and changes in the form and details of the device or process illustrated may be made by those skilled in the art without departing from the invention. The foregoing description is of the best mode presently contemplated of carrying out the invention. This description is in no way meant to be limiting, but rather should be taken as illustrative of the general principles of the invention. The scope of the invention should be determined with reference to the claims appended hereto.